| | |
|---|---|
| Title | Resource scaling effects on MPP performance: The STAP benchmark implications |
| Author(s) | Hwang, K; Wang, C; Wang, CL; Xu, Z |
| Citation | Ieee Transactions On Parallel And Distributed Systems, 1999, v. 10 n. 5, p. 509-527 |
| Issued Date | 1999 |
| URL | http://hdl.handle.net/10722/42822 |
| Rights | Creative Commons: Attribution 3.0 Hong Kong License |

# Resource Scaling Effects on MPP Performance: The STAP Benchmark Implications

Kai Hwang, *Fellow*, *IEEE*, Choming Wang,
Cho-Li Wang, *Member*, *IEEE*, and Zhiwei Xu

**Abstract**—Presently, *massively parallel processors* (MPPs) are available only in a few commercial models. A sequence of three ASCI Teraflops MPPs has appeared before the new millenium. This paper evaluates six MPP systems through STAP benchmark experiments. The STAP is a radar signal processing benchmark which exploits regularly structured SPMD data parallelism. We reveal the resource scaling effects on MPP performance along orthogonal dimensions *of machine size, processor speed, memory capacity, messaging latency,* and *network bandwidth.* We show how to achieve balanced resources scaling against enlarged workload (problem size). Among three commercial MPPs, the IBM SP2 shows the highest speed and efficiency, attributed to its well-designed network with middleware support for single system image. The Cray T3D demonstrates a high network bandwidth with a good NUMA memory hierarchy. The Intel Paragon trails far behind due to slow processors used and excessive latency experienced in passing messages. Our analysis projects the lowest STAP speed on the ASCI Red, compared with the projected speed of two ASCI Blue machines. This is attributed to slow processors used in ASCI Red and the mismatch between its hardware and software. The Blue Pacific shows the highest potential to deliver scalable performance up to thousands of nodes. The Blue Mountain is designed to have the highest network bandwidth. Our results suggest a limit on the scalability of the *distributed shared-memory* (DSM) architecture adopted in Blue Mountain. The scaling model offers a quantitative method to match resource scaling with problem scaling to yield a truly scalable performance. The model helps MPP designers optimize the processors, memory, network, and I/O subsystems of an MPP. For MPP users, the scaling results can be applied to partition a large workload for SPMD execution or to minimize the software overhead in collective communication or remote memory update operations. Finally, our scaling model is assessed to evaluate MPPs with benchmarks other than STAP.

**Index Terms**—Massively parallel processors, SPMD parallelism, ASCI program, STAP benchmark, phase-parallel model, latency and bandwidth, scalability analysis, supercomputer performance.

## 1 Introduction

The MPP performance is attributed to both machine architecture and program behavior. Resource scaling responds directly to projected advances in hardware, software, and system integration. To balance the scaling process, resource scaling must catch up with the scaling in problem size and workload. Scaling determines the smallest machine size or the minimum resources needed to meet the speed requirement of a given application. One can use scaling experiments to predict MPP performance, as well as to guide the scalable design of future Teraflops MPPs.

In Linpack [12], the memory bandwidth is more critical to achieving high performance. In the past, scalability studies were concentrated only in the scaling of problem size [13], [24], [31]. Our study considers the other side of the coin, namely resources scaling to match the scaled workload in future MPP applications. The scaling model is first calibrated with STAP benchmark results on three commercial MPPs. We then apply the model for resource scaling on three exploratory ASCI machines. Holt et al. [15] have suggested a *controlled occupancy* of shared resources, especially for low-latency communication. We will address the same issue through the balancing of growth rates on various resource subsystems in an MPP.

The development of MPP systems followed an evolutionary pattern. For example, the Cray T3E is upgraded from T3D using a streamlined memory technology plus E-Registers for remote memory access [28]. The SGI/Cray Origin 2000 [20], [27] has benefited from the prototype experiences of Stanford DASH multiprocessor project [21]. The demand for even higher performance has never ended. The SP2, Paragon, and T3D all use distributed memory architecture to enable the scalability [18]. Operational parameters of three commercial MPPs are summarized in Table 1.

We start with a benchmark performance evaluation of Cray T3D, IBM SP2, and Intel Paragon. Our experiments are based on the STAP (*Space-Time Adaptive Processing*) benchmark suite for radar signal processing [6]. STAP radar is used to identify real targets from decoys in a sophisticated air defense system. In a STAP application, a backend parallel computer is desired to process a huge volume of reflected radar signals in real time [11]. MPPs are designed to satisfy this computational need [18].

The STAP benchmark suite has become popular among real-time signal processing and radar array processing communities [5], [6], [11], [19]. This paper focuses on how to

- *K. Hwang and C. Wang are with the Deptartment of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089. E-mail: kaihwang@ceng.usc.edu.*
- *C.-L. Wang is with the Department of Computer Science and Information Systems, University of Hong Kong. E-mail: clwang@csis.hku.hk.*
- *Z. Xu is with the National Center for Intelligent Computing Systems, Chinese Academy of Sciences, Beijing, China. E-mail: zxu@apple.ncic.ac.cn.*

*Manuscript received 5 Jan. 1998; revised 14 Aug. 1998.*
*For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 106106.*

TABLE 1
Operational Parameters of Three Commercial MPPs [18]

| Features | IBM SP2 | Cray T3D | Intel Paragon |
|---|---|---|---|
| Configuration and machine site | 256 nodes at MHPCC in Maui | 128 PEs at Cray Eagan Center | 128 nodes at SDSC in San Diego |
| Processor model and peak speed | POWER2, 67 MHz, 266 Mflop/s | Alpha 21064, 150 MHz, 150 Mflop/s | Intel i860, 50 MHz, 100 Mflop/s |
| Latency and bandwidth | 39 μs, 40 MB/s | 2 μs, 150 MB/s | 30 μs, 175 MB/s |
| Aggregate bandwidth | 0.818 GB/s | 1.745 GB/s | 0.879 GB/s |
| MPI implementation | MPICH | CRI/EPCC MPI MPICH | MPICH |

TABLE 2
Summary of ASCI Machine Design Options

| Feature | Intel/SNL ASCI Red[4] | IBM/LLNL Blue Pacific[3] | SGI/LANL Blue Mountain[2] |
|---|---|---|---|
| Processor selection | 200-MHz Pentium Pro with 200 Mflop/s | 500 MHz POWER3 with 800 Mflop/s | SN1 processor with 1 Gflop/s |
| MPP Architecture | Message-passing with distributed memory | Cluster of SMPs with distributed memory | Cluster of SMPs with distributed shared memory |
| No. of processors | 9216 | 4096 | 3072 |
| Peak speed | 1.8 Tflop/s | 3.2 Tflop/s | Over 3 Tflop/s |
| Memory capacity | 594 GB | 2.5 TB | 500 GB |
| RAID Disk | 1 TB | 75 TB | 75 TB |
| Link bandwidth | 800 MB/s | 800 MB/s | 1560 MB/s |
| Available date | June 1997 | Dec. 1998 | Dec. 1998 |

map the partitioned STAP computations for scaled workload and resources. Previously, we reported the STAP performance on SP2 [19], the MPI performance [17], [36], and earlier STAP benchmark results on three commercial MPPs in [33], [35]. Some scalable STAP algorithms are given in [5], [6], [11].

In this study, we focus on the resources scaling effects of the *machine size*, *processor speed*, *message-passing latency*, and *aggregate communication bandwidth* [10], [22]. Our model strives to provide a close match of MPP architecture with the behavior of regularly structured problems. Our STAP experiments were conducted with a nominal radar data set running on current machines with up to 256 nodes. For future MPPs with thousands of nodes, we scale the input workload with a much enlarged data set corresponding to a maximal STAP radar configuration projected for the future.

In 1994, the U.S. Department of Energy launched the *Accelerated Strategic Computing Initiative* (ASCI) program, a 10-year, $1-billion program to build Terflops supercomputer systems [8], [9]. The ASCI program develops MPP systems to replace nuclear weapons testing with three-dimensional numerical simulations. The goal of the ASCI program is to deploy a 1-Tflop/s system by 1996, a 10- to 30-Tflop/s system around year 2000, and a 100-Tflop/s system by 2004. These scalable MPPs, being separately developed by IBM, Intel, and SGI/Cray with three U.S. National Laboratories, are summarized in Table 2.

The Intel/Sandia ASCI Red [4], [23] scales from the mesh interconnect in Paragon. Among the top 500 fastest computers rated in mid-1998, this machine was ranked the first with a 1.338 Teraflops in maximal Linpack speed. Both ASCI Blue machines are cluster-structured using faster floating-point microprocessors. The construction and testing

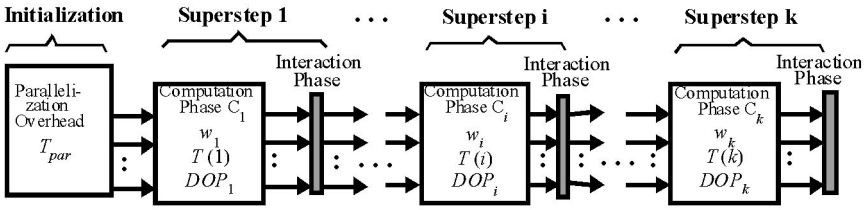



Fig. 1. Phase parallel model for exploiting SPMD data parallelism.

of these two Blue machines are yet to be completed. The IBM/LLNL Blue Pacific [3] absorbs all positive features from the IBM SP Series. The SGI/LANL Blue Mountain [2] is expected to improve from the SGI/Cray Origin 2000, a scalable CC-NUMA (*cache coherent non-uniform memory access*) machine, to a DSM cluster of SMPs (*symmetric multiprocessors*) [18]. The 1-Gflop/s speed of SN1 is needed to achieve a peak speed of 3.0+ Teraflops in the Blue Mountain system.

The machine size, processor speed, latency, and bandwidth are scaled according to the specifications released by the ASCI program. The link bandwidth of Blue Pacific is extrapolated from that of the IBM SP Series. The remaining entries in Table 2 are based on specifications of the three ASCI machines [2], [3], [4]. Our scaling study is applied to two STAP workloads with respect to a small and a large radar configuration. The assessment is based on the past growth and projected future trends in processor speed, network bandwidth, and message-passing latency.

MPP designers and users demand quantitative data to reveal the resource scaling effects on performance. Our study provides partial answers to these questions. Our results reinforce Gustafson's belief [13] that scaled workload will uphold higher system efficiency. However, scaled problems may increase both computational workload and communication overhead at the same time. The balance between the two is hinged on how to reduce the *communication-to-computation ratio* (CCR) in user programs. We will discuss all of these issues in subsequent sections.

The rest of the paper is organized as follows: In Section 2, we present the phase-parallel model for SPMD (*single program and multiple data streams*) data parallelism. Section 3 characterizes the STAP benchmark workload, where the STAP parallelization strategy is outlined. The STAP performance on T3D, SP2, and Paragon is presented in Section 4. The scaling effects on T3D, SP2, and Paragon are given in Section 5. Latency and bandwidth of ASCI machines are assessed in Section 6. The scaling effects of these ASCI platforms are reported in Section 7. Finally, we summarize the research findings and comment on their applicability and limitations. Throughout the paper, we assess six MPP architectures and consider both hardware and software requirements to achieve scalable performance.

## 2 Phase-Parallel Scaling Model

Xu and Hwang [35] have developed a *phase parallel model* for exploiting parallelism in SPMD programs. This scaling model is refined from the BSP model by Valiant [32]. A typical SPMD program is executed in a cascade of *supersteps*, as shown in Fig. 1. Each superstep consists of a *computation phase* followed by an *interaction (communication) phase*. All operations in a superstep must finish before the next superstep begins execution.

The *parallelization phase* lumps all initialization tasks before entering the very first superstep. The overhead incurred with this phase covers the time needed for process creation, termination, grouping, regrouping, etc. The *computation phase* consists of multiple processes to be executed in parallel using local data sets preloaded to the processing nodes. The *interaction phase* executes all forms of communication operations, either point-to-point or collective communications. Massive parallelism exists at both computation and communication phases.

The parallelization adds to the system overhead, denoted as $T_{par}$. Each computation phase $C_i$ is allocated with a computational *workload* of $w_i$ (in Gflop). $T(i)$ denotes the time to execute the $w_i$ workload on one node at phase $C_i$. Thus, the *total sequential time*, $T_1 = \sum_{1 \le i \le k} T(i)$, is the time to execute the entire program on one node, where $k$ is the number of supersteps. *The degree of parallelism*, $DOP_i$ for superstep $i$, denotes the number of parallel computations available at phase $C_i$.

In the interaction phase, collective communication is performed, such as *broadcast, total exchange, barrier synchronization*, or *reduction* operations. A superstep may contain a single computation phase, or just a parallelization phase, or two phases of a computation followed by an interaction. Dividing the program into supersteps enables massive parallelism since the sequential bottleneck is moved to the parallelization phase or to the interaction phase.

Let $T_{comp}$, $T_{comm}$, and $T_{par}$ be, respectively, the *total computation time*, *communication time*, and *parallelization overhead* of an SPMD program. The *parallel time* $T_p$ to execute a program on $p$ processors is equal to the sum: $T_p = T_{comp} + T_{comm} + T_{par}$. When a phase is executed on $p$ processors, where $1 \le p \le DOP_i$, the execution time at phase $C_i$ is simply $T(i)/p$. The total workload equals the sum of workloads in $k$ supersteps. $W = w_1 + w_2 + \ldots + w_k$. Several parallel performance metrics are summarized in Table 3.

However, there are other cases where $p > DOP_i$. Therefore, the actual $T_p(i)$ is calculated by dividing $T(i)$ by the $Min(DOP_i, p)$. Thus, the total *computation time* $T_{comp}$ is expressed as follows:

$$T_{comp} = \sum_{1 \le i \le k} \frac{T(i)}{Min(DOP_{i,p})}. \quad (1)$$

TABLE 3
Performance Metrics in Using the Phase Parallel Model

| Notation | Terminology | Definition |
|---|---|---|
| $T_1$ | Sequential time ($k$= No. of phases) | $T_1 = \Sigma_{1 \le i \le k} T(i)$ |
| $T_p$ | Parallel execution time | $T_p = \sum_{1 \le i \le k} \frac{T(i)}{Min\,(DOP_{i,p})} + T_{par} + T_{comm}$ |
| $W$ | Total workload or total flop count | $W = w_1 + w_2 + \quad + w_k$ |
| $V_p$ | Sustained speed | $V_p = W / T_p$ |
| $S_p$ | System speedup | $S_p = T_1 / T_p$ |
| $E_p$ | System efficiency | $E_p = V_p / (p\, V_{peak})$ |

Different communication operations may require different amounts of time to accomplish, as we have reported in [17]. The time of a typical communication operation is calculated by:

$$T_{comp} = t_o(p) + \frac{m}{R_\infty(p)} = t_o(p) + m \cdot t_c(p), \qquad (2)$$

where $m$ is the *message length* in byte, $t_o(p)$ is the *startup latency*, and $R_\infty(p)$ is the *asymptotic bandwidth*. The inverse $t_c(p) = 1/R_\infty(p)$ is sometimes called the *per-byte time*. Both startup latency and asymptotic bandwidth are functions of the machine size $p$, independent of the message size $m$.

Our benchmark experience shows that the parallelism overhead $T_{par}$ is one or two orders of magnitude lower than the computation time $T_{comp}$ or the communication overhead $T_{comm}$. Thus, $T_{par}$ can be ignored here without loss of much accuracy in the timing analysis. Combining (1) and (2), we obtain the following $T_p$ expression:

$$T_p = \sum_{1 \le i \le k} \frac{T(i)}{Min(DOP_{i,p})} + t_o(p) + \frac{m}{R_\infty(p)}. \qquad (3)$$

As shown in Table 3, the *sustained speed* $V_p$ is obtained by dividing the total workload $W$ by $T_p$. The *system speedup* is computed by the ratio $S_p = T_1/T_p$. Let $V_{peak}$ be the peak speed of a single processor. The *system efficiency* is the ratio of the sustained speed to the peak system speed, defined by $E_p = V_p/(pV_{peak})$.

The *critical path*, $T_\infty$, is defined as the theoretical execution time using an infinite number of processors without worry about any overhead.

$$T_\infty = \sum_{1 \le i \le k} \frac{T(i)}{(DOP_i)}. \qquad (4)$$

The sustained speed $V_p$ is upper bounded by a *maximum speed* defined by $V_\infty = W/T_\infty$. The smallest $p$ to achieve $T_p = T_\infty$ is called the *maximum parallelism*, denoted by $N_{max}$. Using more than $N_{max}$ nodes will not reduce the execution time further. Thus, one can define $N_{max} = \mathrm{MAX}_{1 \le i \le k}(DOP_i)$.

The *average parallelism* is defined as $T_1/T_\infty$, which provides an upper bound on the system speedup, that is $S_pT_1/T_\infty$. Brent has proven [7] that the parallel time $T_p$ is bounded by the following inequality:

$$T_1/p \le T_p < T_1/p + T_\infty.$$

Thus, the following bound on $T_p$ is obtained:

$$Max(T_1/p, T_\infty) \le T_p < T_1/p + T_\infty. \qquad (5)$$

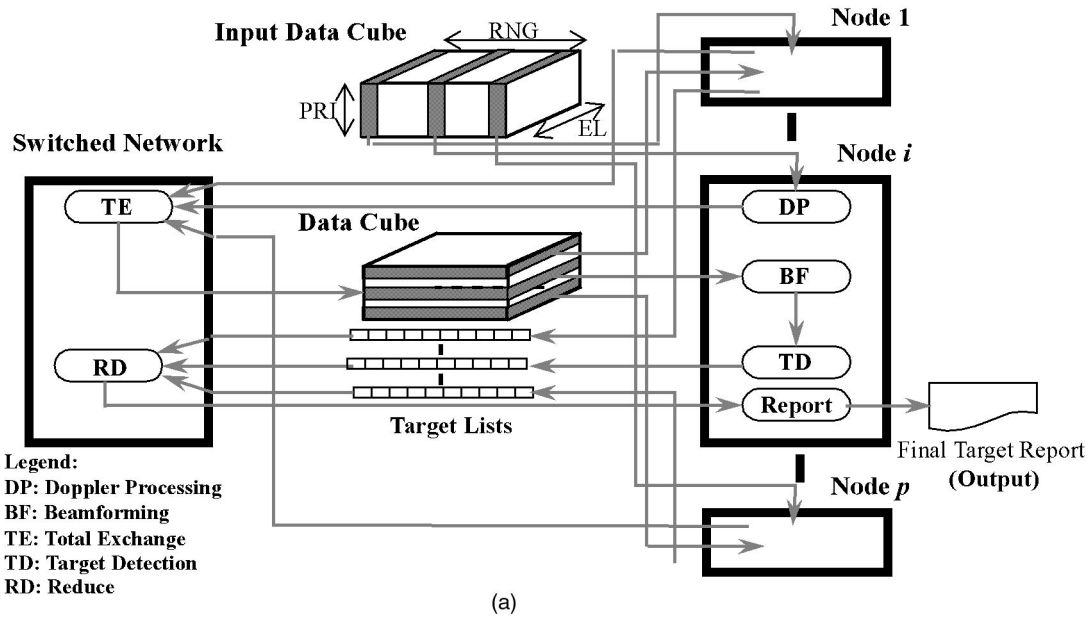
The above performance bounds are very useful to estimate the limits of a given MPP. Equation 2 quantifies the communication overhead. Equation 3 is used to compute the parallel execution time. The critical path and performance bounds can then be easily computed using (4) and (5).

## 3 The STAP Benchmark and Workload

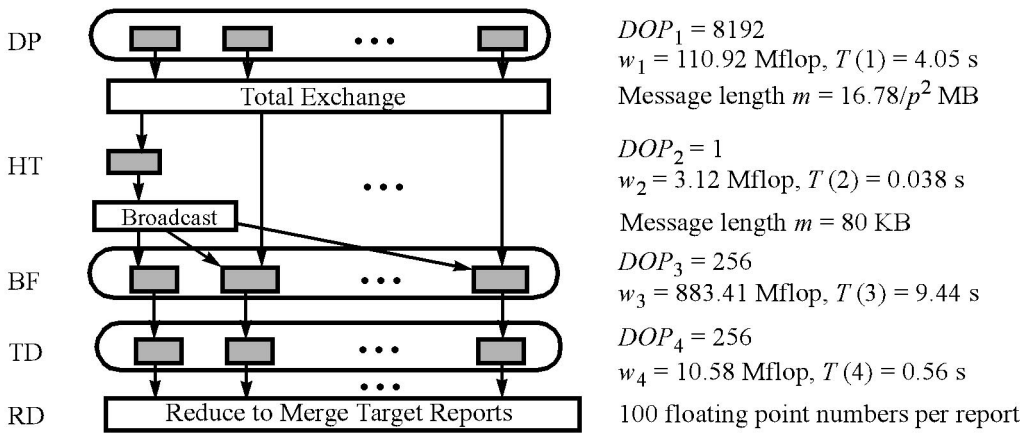
The STAP benchmark was originally developed by MIT Lincoln Laboratory in sequential C code for adaptive radar signal processing on workstations. Our group parallelized the STAP benchmark suite using MPI in five parallel programs. We characterize these programs and the workload and show how to distribute the workload among multiple nodes evenly. The STAP programs consist of some kernel routines, which are often used in signal processing applications.

### 3.1 STAP Benchmark Characteristics

The STAP benchmark suite evaluates MPP by exploiting massive data parallelism. The input radar signals are structured as a 3D datacube, as shown in Fig. 2a. The radar signal datacube changes rapidly with time, corresponding to the continuous generation of radar beams. Therefore, the successive processing of a sequence of datacubes must be done in real time within the PRI (*pulse repetition interval*). The backend computer processes the radar datacube and generate a target list among many targets or decoys detected.




(a)




(b)

Fig. 2. Parallelization of a typical STAP benchmark program (APT). (a) Dataflow in parallel STAP execution. (b) Parallel execution of the APT code on IBM SP2.

The benchmark requires performing $10^{10}$ to $10^{14}$ flop (*floating point operations*) over a huge data set of 0.1 GB to a few hundred GB in the fraction of a second [6], [11]. This translates to a speed of tens of Gflop/s to 100 Tflop/s in real-time applications. We consider two problem sizes in STAP benchmarking experiments: the *small workload* versus the *large workload*. The small workload has a input data set which is bounded by the local memory of the tested MPP node based on current technology. This small workload is applied in Table 4, corresponding to a smaller radar configuration with fewer antenna elements, shorter range gates, etc. [6].

The large workload corresponds to a much bigger radar configuration with a total computational workload of 33.26 Tflop to be performed and a total message size of 3.28 GB for a *total exchange* operation. We shall consider the scaled large workload in Sections 5 and 6 when the ASCI machines are evaluated. The *total workload* $W$ is measured by the total flop count in the program. The STAP is a 32-bit floating-point benchmark. The single-precision executes faster and only uses half of the bandwidth of typical scientific simulation programs that use double precision.

The actual flop count in a parallel program may vary due to communication and parallelization overheads experienced. The workload of the STAP program depends on the size of its input datacube. The *total message length* shows the number of bytes to be communicated. For a fixed-size problem, the total message length is fixed. The average parallelism in STAP programs is achievable on all commercial machines we have tested. We measure the value of $T_\infty$ by the flop count, such as Mflops or Gflops in Table 4, along the critical path.

The CCR shows the average message size communicated by unit of floating-point computation performed, denoted

TABLE 4
Workload Characteristics in STAP Benchmark

| STAP Program | Input Data Size (MB) | Workload W (Gflop) | Average Parallelism | Critical Path (Mflop) | CCR (KB per Mflop) |
|---|---|---|---|---|---|
| APT | 16.78 | 1.008 | 123 | 8.19 | 16.64 |
| HO-PD | 50.33 | 11.22 | 227 | 49.35 | 4.49 |
| Bm-Stag | 41.60 | 2.105 | 325 | 6.47 | 38.91 |
| El-Stag | 36.86 | 1.453 | 89 | 16.40 | 67.68 |
| GEN | 98.30 | 4.133 | 84 | 49.27 | 23.79 |

as KB / Mflop. The HO-PD is computation intensive, hence, its CCR is lower, only 4.49 KB/Mflop, than the others in Table 4. The El-Stag has the highest CCR among all five STAP programs, meaning that it requires to pass more messages than other programs. The value of the CCR depends on program behavior, independent of machine characteristics or runtime conditions.

The STAP benchmark results enabled us to quantify communication overhead in existing MPPs, as well as to project them for the ASCI machines. The scaling results will be particularly useful in converting the exploratory ASCI machines to commercial models in the future. The commercial models must be built with the latest technology [8]. We chose a nominal radar configuration for the STAP benchmark on existing commercial MPPs. For the ASCI machines, the much enlarged workload will be applied.

### 3.2 SPMD Node Program and Internode Communications

Each data element in the signal cube is a complex number (two floating-point numbers). All nodes execute the same program. Each node program consists of three computation steps: *Doppler processing* (DP), *beamforming* (BF), and *target detection* (TD), as shown in Fig. 2b. The *total exchange*, *broadcast*, and *reduction* are needed in collective communications. The input data cube is partitioned into $p$ equal slices along the *range-gate* (RNG) dimension. Each node takes one slice as its input data subcube.

In the Doppler-processing step, all nodes execute 8,192 FFT routines on their allocated data slices simultaneously. After Doppler processing, every node must exchange their results of FFT with every other nodes via a *total exchange* operation, which involves $O(p^2)$ pairs of point-to-point communications. The data distribution after *total* exchange is represented by the horizontally sliced data cube, which is partitioned into $p$ equal slices along the PRI dimension. The *total message length* is the sum of all messages among all node pairs involved. Thus, the individual message length equals the total message size 16.78 MB divided by $p^2$ messages over $p$ nodes.

The *Householder Transform* (HT) is a sequential step performed on a single node. The HT results are broadcast to all nodes. The next phase is to perform the beamforming operations on all nodes. In the final *Reduction* (RD) phase, each node generates a partial target list, which will be merged to form a single target list as the output to the user.

The granularity varies among the five benchmark programs. The HO-PD is the most computation-intensive one with a total flop count of near 11.2 Gflop for a nominal small workload. On a large radar configuration, this workload could increase to 33.26 Tflop, to be discussed in Sections 6 and 7. The STAP benchmark needs to perform *point-to-point*, *broadcast*, *reduce*, and *total exchange* communications. The broadcast is done with a logarithmic algorithm [17]. The fast delivery of long message relies more on the network bandwidth. The short message is often slowed down by the start-up latency experienced.

## 4 Measured Stap Benchmark Results

Benchmark experimental results of thousands of parallel STAP run on the T3D, SP2, and Paragon are reported below. The memory, I/O, and communication bandwidth requirements are revealed. Architectural implications of STAP benchmark results are given.

### 4.1 Measured Performance and Bottlenecks

First, we present the sustained speed of three current MPPs and then identify their performance bottlenecks in executing the STAP programs on these machines. The sustained speed is the raw speed measured in a benchmark run. In real-time applications, such as the STAP, the sustained speed is the most important one, instead of the normalized speed with respect to a reference machine. To some extent, the relative speed is reflected by the system efficiency. These speed measures are reported and analyzed below.

#### 4.1.1 Sustained System Speed

Using the critical path values in Table 4 and (5), we obtain the speed bounds on three MPPs: SP2 within 9.34-11.0 Gflops, T3D within 3.89-4.39 Gflops, and paragon within 2.63-2.98 Gflops, if 128 nodes are used. In reality, the sustained speed is much lower, as shown in Fig. 3, varying with machine sizes and different platforms.

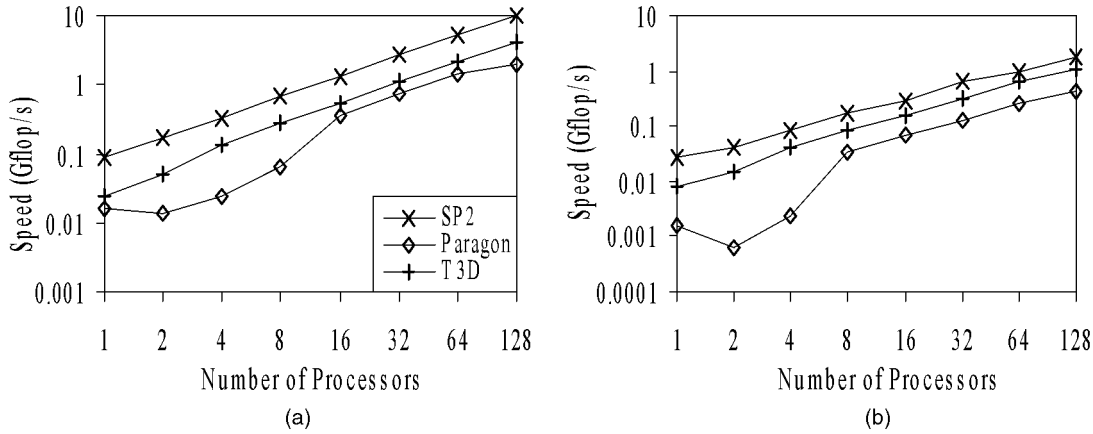



Fig. 3. Sustained speed of three commercial MPPs.

Based on our measurements, none of the commercial machines has exceeded 10 Gflop/s speed in executing the STAP programs up to 128 nodes. The best case is the SP2, which achieved a 9.8 Gflops speed with 128 nodes, as shown in Fig. 3a. In general, the SP2 achieved the highest sustained speed, followed by the T3D, and the slowest is the Paragon. This is true for all 15 machine-program combinations we have evaluated.

Only the speed results of two extreme programs are illustrated in Fig. 3. The HO-PD program is a computational intensive program with a workload of 11.2 Gflop (Table 4) and a nominal input data cube of 50.33 MB. It requires communicating a total of 51.1 MB of messages. Hence, its CCR is rather low as 0.049 Byte/flop. The EL-Stag program is communication-intensive with the largest CCR rate in Table 4.

On a 64-processor system, the SP2, T3D, and Paragon achieved the speeds of 5.2, 2.1, and 1.4 Gflop/s, respectively. With 128 nodes, the SP2 achieved the highest speed of 9.8 Gflop/s. A 128-node Paragon achieved 2.0 Gflop/s speed in executing the HO-PD program. The performance dip of Paragon with eight nodes or less is due to the *disk swap* delay (DSD) effect reported in [33]. The disk swap delay is caused by the collective communications operations involving a large number of messages to be exchanged simultaneously. Obviously, there is no communication cost on a single-node machine.

The speed results of the EL-Stag program are shown in Fig. 3b. Both the Paragon and T3D experienced a speed dip on a small configuration of 16 or less nodes. This is again due to the DSD effect. This program has a huge communication overload because it needs to handle collective messages of 98.3 MB all in one phase (see Table 4). The communication time in the EL-Stag program occupies a large percentage in the total execution time. This slows down the speed of EL-Stag program by 10 times, compared with that of the HO-PD program.

### 4.1.2 Time Breakdown

To identify the bottleneck, the HO-PD execution time is plotted in Fig. 4a for the small machines of no more than eight nodes and in Fig. 4b for medium-sized machines from 32 to 128 nodes. The disk swap time appears only in small configurations of Paragon. The major bottleneck in a small MPP configuration is resulted from the disk access penalties as shown on top of bars in Fig. 4a. For example, on a 4-node Paragon, the disk access time is 331s, about 73 percent of the overall execution time.

The Paragon shows the worst disk swap penalty. The SP2 has no disk problem at all. The T3D has this problem only up to four nodes. This is due to the fact that the local memory in each Paragon was too small (16-32 MB) to handle the large data set we have in STAP benchmarks. The SP2 uses 256 MB per node to avoid the disk swap problem. The communication time is very small in small MPPs.

On medium-sized machines, the beamforming (BF) consumes the most computation time, as sown in the white portions of the bars in Fig. 4b. The communication time shows in black sections on top of the bars. The disk access time is reduced to zero because each subdivided data slice can fit the local memory entirely. However, the reduction in communication time is much slower than that of the BF time because HO-PD has the lowest CCR among the five. As the machine size increases, the major reduction in execution time is found in the parallel execution of the BF phase rather than the DP phase.

### 4.1.3 System Efficiency

The *system efficiency* is defined as the ratio of the sustained system speed to the peak system speed. The ranges of system efficiency of three commercial MPPs are shown in Fig. 5, considering all five STAP programs running on four MPP sizes. The SP2 achieved the highest efficiency of around 30 percent. However, the SP2 has a wide efficiency range, between 2 percent and 33 percent. The Paragon has the lowest efficiency of at most 17 percent. The T3D has very stable efficiency rate of 22 percent to 25 percent for all machine sizes.

Compared with the NAS and Linpack reports, these efficiencies are not considered that low. This means the parallelization of the STAP code was fairly successful on the MPPs. On the other hand, all three machines have shown less than 5 percent efficiency at the low end. This implies that some of the STAP benchmark codes, with large sequential bottlenecks, cannot take advantage of the massive parallelism provided by the MPPs.

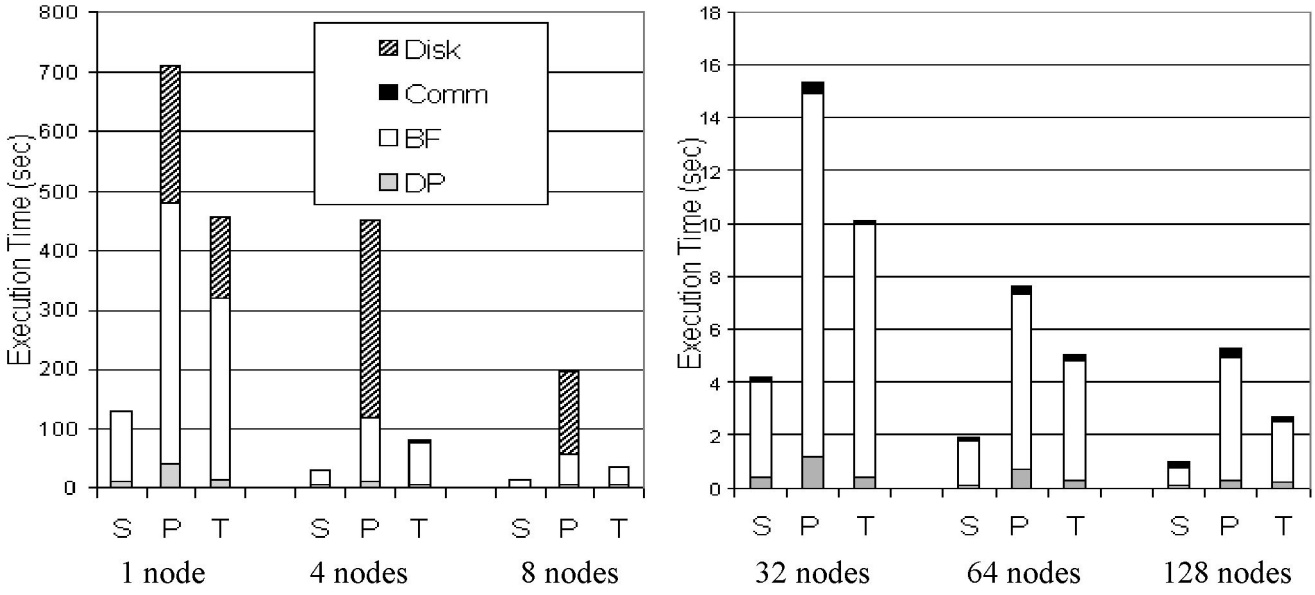



Fig. 4. Breakdown of the HO-PD execution time on three MPPs coded as S for SP2, P for Paragon, and T for T3D: (a) Small machines, (b) medium-size machines.

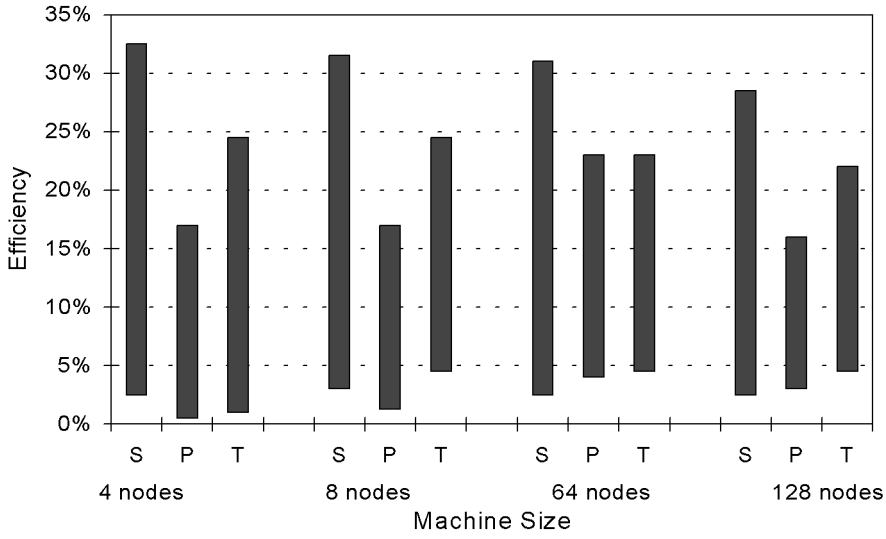



Fig. 5. System efficiency in executing STAP programs on three MPPs. (Captions: S = SP2, P = Paragon, and T = T3D).

The system efficiency decreases with increasing machine size in STAP benchmark experiments. In a relative sense, the system efficiency reflects the normalized speed performance because different-speed processors are used in the MPPs. Both sustained speed and normalized speed (efficiency) are important in the evaluation of MPP performance. The sustained speed is more useful to real-time applications like STAP. The normalized speed gives a fairer comparison among different MPP architectures.

### 4.2 Memory, I/O, and Communication Requirements

We discuss below the STAP results and their architectural implications on the memory, I/O, and communication requirements in commercial MPP systems.

#### 4.2.1 Memory Requirement

A key parameter for parallel execution is the physical memory capacity per node. This parameter also affects the processor selection in the node design and its physical address space. The per-node memory requirements for the HO-PD program are shown in Fig. 6a. The diamond-curve corresponds to data memory for each node program. The squares refer to the program memory. The total curve is the sum of the two.

The memory requirement per node is estimated by $C_1/p + C_2$, where $p$ is the number of nodes allocated and $C_1$ and $C_2$ are two program dependent constants. The values of $C_1$ and $C_2$ depend on the data slices, memory for temporary I/O, and communication message buffers, etc. The divided node program/data sets demand different amounts of memory for various machine sizes. For larger machines, the total data memory required approaches $C_2$, which accounts

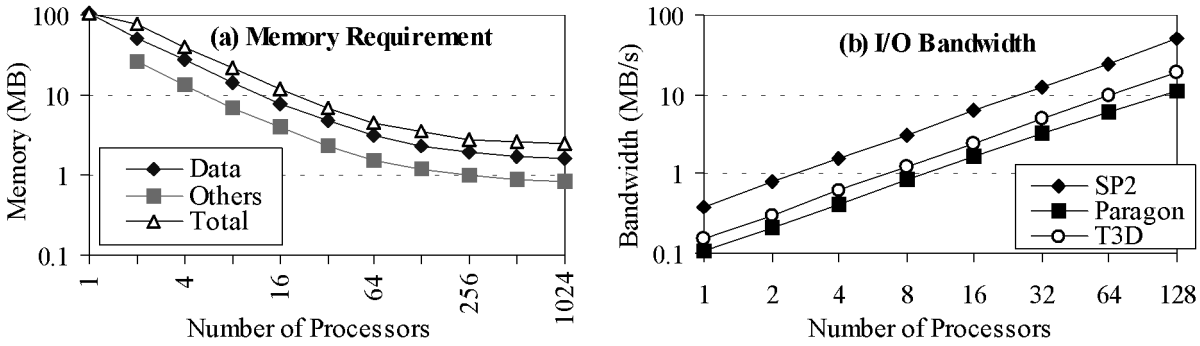



Fig. 6. Per-node memory and I/O requirements of the HO-PD program.

for the data size and sequential and control portions of the node program. For example, the HO-PD program requires $(102.8/p + 1.5)$ MB data memory, and the program memory uses $(50.3/p + 0.79)$ MB using data from Table 4.

Most STAP benchmarks have large problem sizes and need more memory than what can be accommodated by a single node on most MPPs in use today. However, the required memory per node decreases almost linearly when more nodes are added. The memory buffer used for message passing is just a small percentage of the total memory. For instance, for the HO-PD running on a Paragon with 16 MB per-node memory, at least 16 nodes must be used. Otherwise, the performance goes down sharply due to excessive page faults.

#### 4.2.2 I/O Requirement

While executing the STAP programs, the successive processing of a sequence of datacubes must be transmitted from the distributed disk arrays or a centralized I/O subsystem to the backend MPP in real time. Thus, the backend computer can process the radar datacube and generate a target list among many targets or decoys detected. The I/O bandwidth requirement depends on the I/O data size, the machine size, and memory/disk hierarchy.

In STAP benchmark, the data size is predetermined for each program. To avoid the I/O bottleneck, the *I/O bandwidth* is defined as the ratio of the datacube size over the total data transfer time from disks to internal memory. Based on the STAP operational requirement, the I/O transfer rate must match with the parallel-processing rate among the nodes. Fig. 6b shows that the I/O bandwidth of the HO-PD benchmark program increases quadratically with respect to the machine size. These I/O requirements suggest at least a 1 GB/s I/O bandwidth when less than 128 processors are used.

The leveling-off of the I/O requirement for an even larger machine is due to the sequential bottleneck in each node program. When the processor speed reaches 1 Gflop/s, the required I/O bandwidth falls within the range of 100 to 500 MB/s. With current disk and RAID technology, sequential I/O (single-disk or single RAID) is enough for small to medium machine sizes (e.g., up 64 nodes). But if the machine size exceeds 64 processors, then parallel I/O is needed to maintain a balanced system performance. With regular I/O access patterns, parallel I/O can access the data file in stride across multiple disks.

#### 4.2.3 Communications Bandwidth

An often asked question is: How fast a communication subsystem is needed in an MPP design? Both *latency* (or startup time) $t_o$ and *asymptotic bandwidth* $R_\infty$ can partially answer this question. A rule of thumb for collective MPI communication is to make it shorter than the computation time [15]. The bandwidth requirement for the El-Stag program is shown in Fig. 7.

El-Stag is the most communication-intensive program in the STAP benchmark suite. The three curves show that the bandwidth increases with respect to the number of processors involved in a collective communication. The communication bandwidths of SP2, T3D, and Paragon in Fig. 7 are in consistency with the speed ranking in Fig. 3b and the I/O bandwidth ranking in Fig. 6. The speed and bandwidth are highly correlated in communication-intensive applications. For SPMD programs with low CCR value (such as the HO-PD program), this correlation may be not necessarily hold.

## 5 Scaling Effects On Commercial MPPs

Scaling can be conducted at various dimensions, such as the *machine size*, *problem size*, *processor speed*, *messaging latency*, and *aggregate* bandwidth of an MPP system. We analyze the scaled performance of T3D, SP2, and Paragon with respect to a fixed workload. In other words, we scale along all resources dimensions, but not the problem size applied to current machines. Increasing the problem size will be studied in the next section when the ASCI design options are assessed.

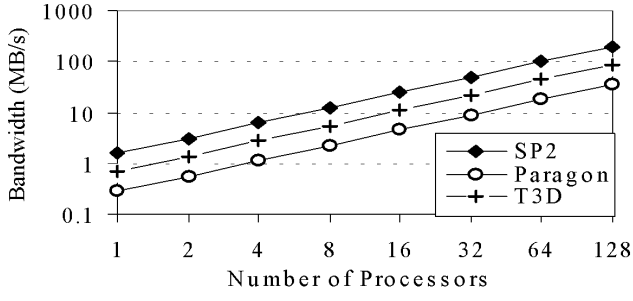



Fig. 7. Communication bandwidth of the El-Stag program.

TABLE 5
Measured Machine Parameters of Paragon, SP2, and T3D in STAP Benchmark Experiments with a Small Workload

| Parameters / Code | | IBM SP2 | Intel Paragon | Cray T3D |
|---|---|---|---|---|
| Sequential time $T_1$ in second | APT | 14.1 s | 58.3 s | 25.5 s |
| | HO-PD | 129.4 s | 482.2 s | 327.6 s |
| | Bm-Stag | 45.7 s | 307.0 s | 98.9 s |
| | El-Stag | 55.1 s | 311.6 s | 124.7 s |
| | GEN | 130.0 s | 478.6 s | 269.3 s |
| Latency $t_o(p)$ for total exchange in μs | | $38.2\,p^{0.908} + 21.4$ | $65.8\,p^{1.153} + 163$ | $117\,p^{1.013} + 59.6$ |
| Asymptotic bandwidth $R_\infty(p)$ for total exchange in MB/s | | $46.8\,p^{0.71} - 41.8$ | $227\,p^{0.297} - 252$ | $26.1\,p^{0.866} + 34.3$ |
| Total Message Length m | APT | 16.78 MB | | |
| | HO-PD | 50.33 MB | | |
| | Bm-Stag | 40.96 MB | | |
| | El-Stag | 32.77 MB | | |
| | GEN | 98.3 MB | | |

Table 5 summarizes important scaling parameters for executing the STAP benchmark programs on three current MPPs. All data entries are based on measurements on the target machines. The latency and bandwidth expressions were derived from our previous work [17]. The *total message size*, $m$, is the sum of all messages communicated in a collective operation.

Using (2), the latency and bandwidth expressions are derived for the *total exchange (all-to-all)* operation in Table 5. This is the most time-consuming collective communication operation. Similar expressions can be also derived for other collective communication operations such as *broadcast* and *reduction* operations mentioned in Fig. 2b. In our previous papers [17], [36], timing experiments and procedures are given to quantify the coefficients and powers in the latency and bandwidth expressions. The details will not be repeated here.

### 5.1 Scaling in Machine Size

The speed performance of STAP benchmark on SP2, Paragon, and T3D is depicted in Fig. 8. We have only considered the scaling up to 1,024 processors. The sustained speed increases steadily as the machine size increases. The El-Stag does not scale well with large machines for its high CCR encountered in Fig. 8a. On the other hand, the computation-intensive HO-PD program scales well in Fig. 8b. In both cases, the SP2 has demonstrated the best speed performance due to the use of a scalable network and fast POWER2 processors.

The Paragon scales better than the T3D for the El-Stag program, but their ranking is reversed in the HO-PD code. The main reason is that i860 in Paragon is much slower than Alpha 21064 in T3D. The T3D scales better for computation-intensive codes. Another limit is the network bandwidth. When the CCR is high in a program like El-Stag, the network bandwidth limits the size scalability. When the computation workload is high (like the HO-PD code), all three MPPs scale well.

Based on 1998 technology, our scaling results imply that a machine size up to 4,169 processors can still improve the HO-PD performance on all three machines. Beyond this limit, the sustained speed begins to fall due to a sharp increase of communication time. In the case of the El-Stag code, the upper limit is lowered to 1,200 processors in size scalability.

The key message being conveyed here is that the size scalability is very sensitive to the CCR in user programs. Among five STAP programs, the scaling limit falls between 968 and 4,196 processors, depending on the program structure. This result confirms the fact that a maximum of 1,024 processors in commercial MPPs was indeed a good choice in 1997. The system cost does not increase linearly with the number of processors used. In other words, cost scaling is equally important as resources scaling. However, cost-effectiveness is a very complex issue which is beyond the scope of this paper.

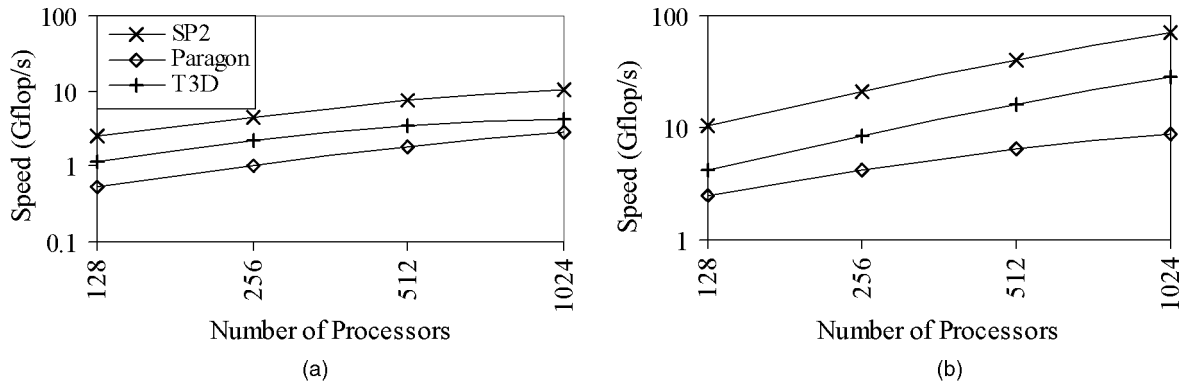



Fig. 8. Effects of increasing machine size on the performanceof two STAP programs on three commercial MPPs.

## 5.2 Scaling in Processor Speed

In general, the system speed improves with faster processors. With 128 processors, we show the speed scaling effects on three commercial machines in Fig. 9. According to Moore's Law [25], the processor speed can improve from 1 Gflop/s to 64 Gflop/s in nine years. With 128 1-Gflop/s processors, the peak would be 128 Gflop/s. However, Fig. 9 shows a sustained speed between 8 and 50 Gflop/s. Using faster processors, the Paragon shows the sharpest increase in system speed.

The T3D shows the slowest increase rate. This is mostly explained by the network scalability of the 2D mesh in Paragon, as compared with the 3D torus in T3D. The SP2 shows some slowdown when the processor speed increases beyond 64 Gflop/s. None of the three machines can scale to 1 Tflop/s performance with processor speed increases to 64 Gflop/s. The system speed reaches a limiting value when the processor speed scales infinitely.

For HO-PD program to reach 90 percent of its limiting speed, the processor speed of SP2, Paragon, and T3D must increase 239, 3,005, and 337 times, respectively. Further increase in the processor speed will boost at most 10 percent of the performance. This result shows the importance of matching the memory latency and network design with the scaling of the processor speed. Overshot in processor speed may create performance bottleneck.

The SP2 reaches 90 percent of the saturated speed when its processor increases in speed by 239 times. The lower the saturation point is, the better a match is expected between the network/memory hierarchy and the processor speed. In this sense, the SP2 processor matches nicely with its switch network characteristics. The Paragon has the poorest match between its subsystems. The T3D sits in the middle between the extreme cases.

## 5.3 Scaling in Latency and Bandwidth

Both latency and bandwidth affect the MPP performance. The effects of reducing latency are rather limited. But the aggregate bandwidth has greater effects on the collective communication, especially when the message length is large. In both cases, the effects become saturated quickly in all three machines.

### 5.3.1 Reducing Communication Latency

With reduced latencies for *total exchange* over 128 processors, the performance of the El-Stag program is depicted in Fig. 10a. The latency accounts for all software overhead to establish the communication links within a communication group and the time required to set up the message buffers and to process the messages. The sustained speed of the El-Stag program is plotted below against a scaled-down communication latency for all three commercial MPPs.

The system speed increases with reduced latency. It quickly reaches a saturation value when the latency becomes sufficiently small. The current latencies for total exchange on SP2, Paragon, and T3D are 3.2, 5.6, and 9.7 ms, respectively. When the latency is reduced to 10 $\mu$s, the speed improves from the current speed by 2 percent on SP2 and Paragon and 5 percent on T3D. In summary, the communication latency controls only less than 5 percent of the STAP speed.

The values of current latency and bandwidth in paragon and T3D have reached their saturated speed range in Fig. 10. The latency is relatively unimportant since parallel STAP programs use collective communication to exchange large amount of data. Hence, the bandwidth requirement is lot more important as shown in Fig. 10b. The darkened dots in Fig. 10 correspond to the latency and bandwidth values measured on commercial machines available in late 1997.

### 5.3.2 Scaling in Bandwidth

The performance of El-Stag program is plotted in Fig. 10b against increasing aggregate bandwidth in three MPPs. Again, we consider the *total exchange*operation over 128 processors. Running the El-Stag code on SP2 is most sensitive to the bandwidth increase. But the scaling effects become quickly saturated as well. The speed increases sharply for the SP2 with increasing bandwidth. The SP2 achieved an aggregate bandwidth of 1.4 GB/s and a sustained speed of 2.5 Gflop/s. Further speed increases becomes saturated as the bandwidth becomes greater.

The T3D achieved 850 MB/s aggregate bandwidth and 1.18 Gflop/s speed. The SP2 and T3D scale within 12 percent and 9 percent of the limiting speed when the bandwidth increases to 1 GB/s. The Paragon achieved a bandwidth of 6.87 GB/s and a low speed of 530 Mflop/s. Scaling bandwidth has almost no effect on the Paragon. For the

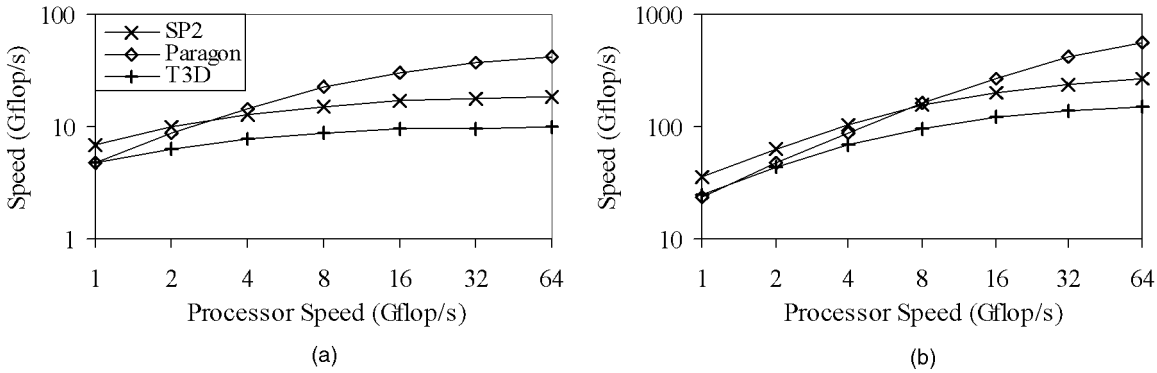



Fig. 9. Effects of increasing processor speed on three commercial MPPs.

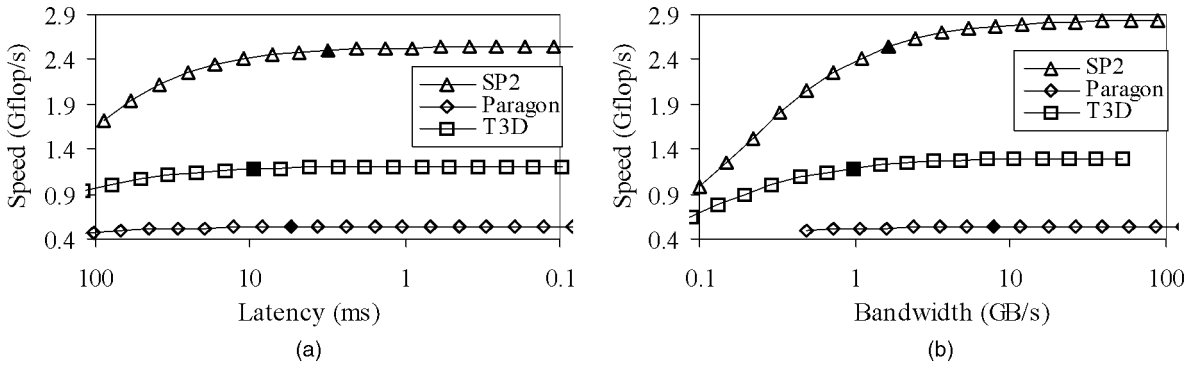



Fig. 10. Effects of reducing latency and increasing bandwidth. (Late 1997 values are highlighted by darkened dots.)

El-Stag program, the bandwidth effects are greater than that for the HO-PD program. With lower bandwidth, increasing processor speed becomes saturated faster. The increase in processor speed must match the bandwidth increase to have a balanced growth.

## 6 Latency and Bandwidth of ASCI Machines

The scaled, large workload cannot be tested on existing MPPs such the T3D, SP2, or Paragon for a memory-bound problem. Because the ASCI machines are much bigger in machine size, memory capacity, and communication bandwidth, we will test them with a much enlarged workload. The large and small workload differs almost 3,000 times in magnitude. For the ASCI machines, latency and bandwidth are mostly unknown because only the Intel/SNL Option Red was built when this paper was written. The other two ASCI design options are still under construction and will not be delivered to the user sites until late 1998 or later.

We use an extrapolation method to estimate the latency and bandwidth values of the ASCI machines. We estimate these *parameters* for possible execution of the HO-PD program on ASCI machines. This estimation is guided by the ASCI specifications reported by Crawford [9]. The common goal of the three ASCI machines is to have a network bandwidth which is linearly scalable. Therefore, the asymptotic bandwidth is mainly affected by the link bandwidth.

Machine parameters for executing the STAP benchmarks on ASCI machines are calibrated in Table 6 for a scaled workload. These parameters are stretched from those in Table 5 with the help from data recently released from Intel TFOPS machine [4], [23], the latest IBM/SP extension [3], and the SGI/Cray Origin 2000 upgrade [2], [27]. All ASCI computers emphasize the architectural scalability with respect to changing workload.

The *sequential time* $T_1$ corresponds to execution on a single node of the ASCI machines. The average time to execute a floating point operation is tied to the peak processor speed. The processor speed for Paragon, SP2, and T3D are 100, 266, and 150 Mflop/s, respectively, and the processor scaling factor for processors used in Option Red (200 Mflop/s), Blue Pacific (800 Mflop/s), and Blue Mountain (1 Gflop/ s), are 2, 3, and 6.67, respectively.

The same reasoning was also applied to estimate the latency expressions in Table 6. Scaling of latency is done by reducing the coefficients in the latency expressions in Table 5 while keeping the exponential terms unchanged. The coefficients are reduced inversely in proportion to the increase of the processor speed, because higher processor speed reduces the software overhead in passing messages.

Scaling of the asymptotic bandwidth is tied to the per-port link bandwidth and bisection bandwidth of the interconnection networks used in the ASCI machines. The link bandwidth of Paragon, SP2, and T3D were given as 175, 40, 150 MB/s, respectively. The corresponding link bandwidths of Option Red, Blue Pacific, and Blue Mountain are given as 800, 800, 1,560 MB/s, as listed in Table 2, respectively. The ASCI Red implemented a 2D mesh stretched from the Paragon [4], [23]. The Blue Pacific

TABLE 6
Machine Parameters Calibrated for Executing STAPBenchmarks on ASCI/MPPs with a Large Workload

| Parameters | Intel/SNL Option Red | IBM/LLNL Blue Pacific | SGI/LANL Blue Mountain |
|---|---|---|---|
| *Sequential time* $T_1$ | 624,043 second | 111,605 second | 127,171second |
| Total message size, $m$ | 3.28 GB | | |
| Total Workload, $W$ | 33.26 Tflop | | |
| Latency $t_o(p)$ for total exchange, (μs) | $32.9\,p^{1.153} + 81.3$ | $12.7\,p^{0.908} + 7.13$ | $17.6\,p^{1.103} + 8.94$ |
| Asymptotic bandwidth $R_\infty(p)$ for total exchange, (MB/s) | $1038\,p^{0.297} - 1152$ | $117\,p^{0.71} - 104$ | $272\,p^{0.866} + 357$ |

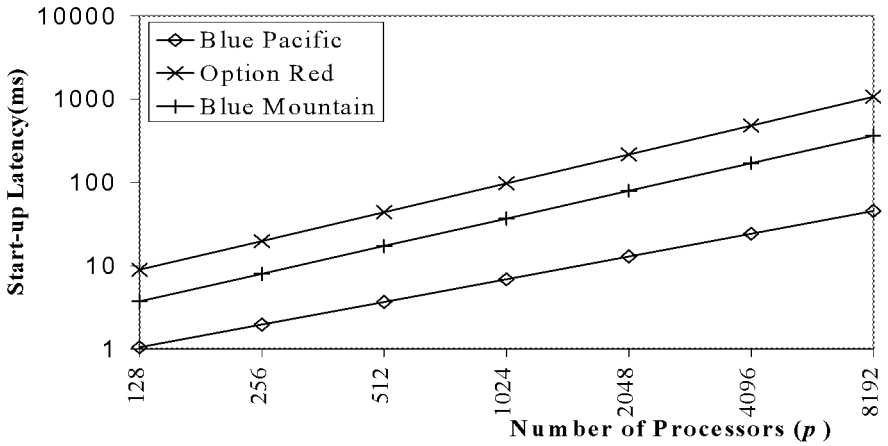



Fig. 11. Projected latency for total exchange on the ASCI/MPPs.

architecture is based on the *High Performance Switch* (HPS) used in SP2 [3]. The Blue Mountain is based on using the SGI/Cray routers and the fat hypercube architecture [2], [20] as built in the Origin 2000.

Our estimation of the latency and bandwidth in the ASCI machines are primarily based on extending data from these predecessor commercial machines. Fig. 11 plots the predicted startup latencies for message passing on the ASCI machines. The latency unit was indicated in *ms*, 1,000 times higher than $\mu$s used to measure the latency in today's commercial computers.

The latency increases slowly in the Blue Pacific design, which has the shortest latency among the three machines. The Blue Pacific supports an affinity property which allocates the nodes within the same SP frame to the same user application. This property can avoid longer latency across the SP2 frame boundary. As the machine size increases from 128 to 8,192 processors, the latency of Blue Pacific increases from 1 ms to 15 ms.

The Blue Mountain is second in latency growth, showing an increase from 5 ms to 100 ms. The ASCI Red is the worst, growing from 10 ms to 1,000 ms on very large machines. This long latency of ASCI Red is caused by the slow Pentium Pro processors used. It is also due to the adoption of long-latency mesh architecture. Fig. 12 estimates the growth of the *asymptotic bandwidth* in three ASCI machines for sufficiently long messages. The Blue Mountain design scores the highest asymptotic bandwidth of 666 GB/s if the machine scales to 8,192 nodes. This design stretches the S2MP nodes and the fat hypercube architecture of the Origin 2000 to the extreme. The 1,560 MB/s link bandwidth of the Blue Mountain design is twice as that of the Blue Pacific design.

Both ASCI Red and Blue Pacific show a poorer asymptotic bandwidth due to the use of interconnection networks with narrow links and smaller bisection bandwidth. The bandwidth of the Blue Pacific outperforms that of ASCI Red for having much higher bisection bandwidth in its network design. In general, a communication-intensive program (such as the EL-Stag in STAP benchmark) demands higher bandwidth requirement, while the computation-intensive one (like the HO-PD) relies on the use of a low-latency network and high-speed processors.

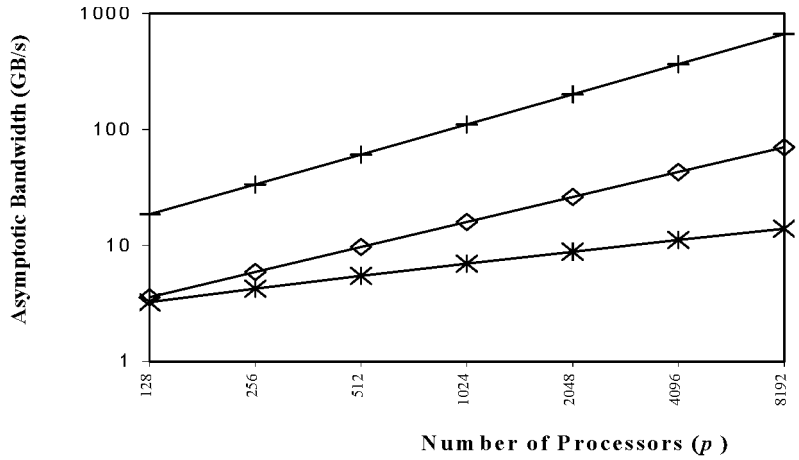



Fig. 12. Asymptotic bandwidth for total exchange on the ASCI machines.

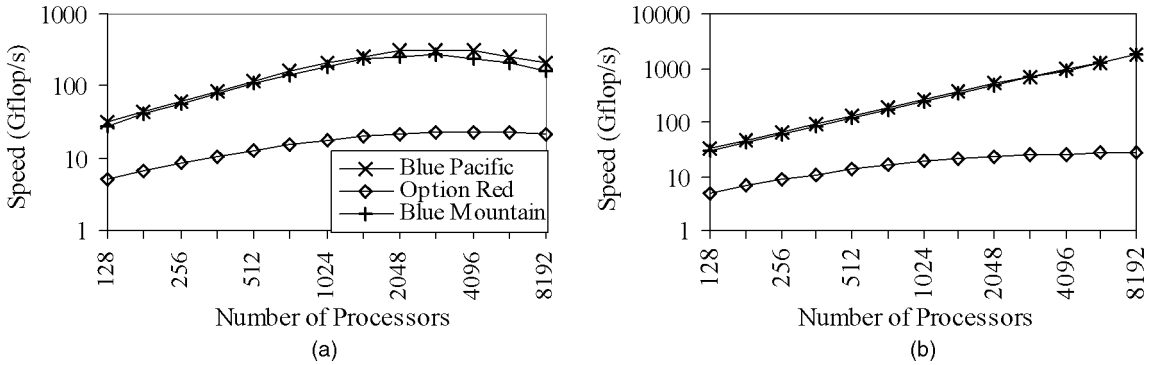



Fig. 13. Effects of machine size on the performance of ASCI machines.

## 7 Scaling in ASCI Machine Design Options

In this section, the scaling effects on machine size, problem size, processor speed, start-up latency, and asymptotic bandwidth are presented separately. Again, cost scaling is not among studies.

### 7.1 Scaling in Machine Size

The machine size is scaled to predict the highest *system speed* achievable by ASCI computers. The speed effects in scaling the machine size from 128 to 8,192 processors are projected on all three ASCI machines in executing the HO-PD program. The *system speed*, $V_p = W/T_p$, is calculated by the time expression for $T_p$. Speed performance covers both small workload (Fig. 13a) and large workload (Fig. 13b).

The parallel execution time $T_p$ is derived from (3). The latency and bandwidth given in Table 6 are used to derive the communication overheads, $T_{comm} = t_o(p) + m/R_\infty(p)$, by varying the machine sizes. Blue Pacific and Blue Mountain are projected to have approximately the same performance as shown by the top curves in Fig. 4. They both outperform the Option Red, mainly due to the use of faster processors.

For the small workload, the maximum speed to run the HO-PD program on ASCI Red, Blue Mountain, and Blue Pacific are 22.6 Gflop/s (with 4,169 processors), 266.8 Gflop/s (with 2,552 processors), and 317.3 Gflop/s (with 2,885 processors). The predicted performance is less than 10 percent of the claimed peak performance of the three ASCI machines. The small workload cannot demonstrate the full computing power designed in the ASCI machines. Beyond the optimal machine size (indicated within the parentheses above), further increase on machine size will result in a decrease of sustained speed. This is due to the offset by the increase of communication overhead, which cancels the benefit from using more processors.

With a large workload, the sustained system speed does show a sublinear speedup in the Blue Pacific and Blue Mountain options. The scaled workload does pay off as claimed in Gustafson's law [13]. Both ASCI Blue machines can easily achieve 1-Tflop/s performance at a machine size exceeding 8,000 processors. As shown in Table 7, there is a dramatic difference in the optimal machine size and the maximum performance between the El-Stag and HO-PD programs.

In the El-Stag code, the maximum speed is 30 Gflop/s for both ASCI Blue machines. The first number in Table 7 is the machine size and the number in parentheses is the maximal speed achieved. The ASCI Red requires a very large machine size to achieve a high speed. This suggests that fast processors must match with faster networks. The optimization in machine size in Table 7 cannot be repeated on other system parameters because faster processors and faster networks are always desired with lower latency and higher bandwidth to yield higher performance.

TABLE 7
The Optimal Machine Sizes of ASCI Machines for Executing Two STAP Codes

| ASCI MPP Platforms | Optimal Machine Size and Speed Performance in parentheses | |
|---|---|---|
| | HO-PD | El-Stag |
| Blue Pacific | 2885 (317.3 Gflop/s) | 1201 (30.1 Gflop/s) |
| ASCI Red | 4169 (22.6 Gflop/s) | 1959 (6.9 Gflop/s) |
| Blue Mountain | 2552 (266.8 Gflop/s) | 946 (28.2 Gflop/s) |

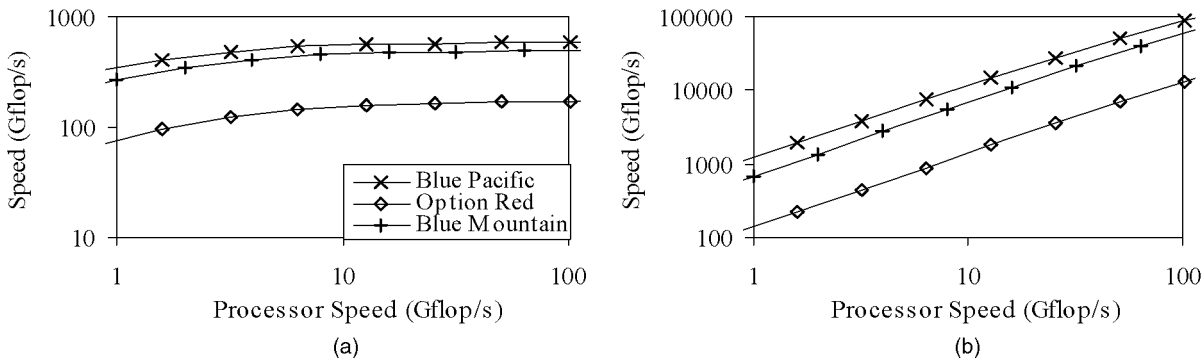



Fig. 14. Effects of processor speed on the ASCI performance.

### 7.2 Scaling in Processor Speed

The projected effects of using even faster processors in three ASCI machines are depicted in Fig. 14. Again, we consider the execution of the HO-PD code with respect to two workload sizes. All ASCI machines can be upgraded with the use of faster processors, but not necessarily in a linear fashion. For a small workload, the speed saturates quickly, as shown in Fig. 14a. For a large workload, almost a linear speedup is observed in Fig. 14b. The Blue Pacific shows a speed exceeding 1 Tflop/s using 1-Gflop/s processors. The large workload is sufficiently large to keep all processors busy.

### 7.3 Effect of Latency and Bandwidth

The performance of the El-Stag program with different communication latency is depicted in Fig. 15a. The scaling of latency reflects the impact on the performance of the ASCI MPP platforms. The system speed increases while decreasing latency, and then saturates to a limiting value quickly.

When the latency is reduced from 1 ms to 0.1 ms, the sustained speed improves 20 percent in the Blue Pacific and Blue Mountain machines. The ASCI Red improves only 2 percent in speed in the same latency range. The speed of the El-Stag program increases with the network bandwidths of ASCI computers, as depicted in Fig. 15b. In all cases, the Option Red shows flat low speed performance. The sustained speed increases with bandwidth, and then gets saturated.

When the bandwidth increases from 10 to 100 GB/s, the speed improves less than 10 percent in the Blue machines. The lower curves show that the Option Red does not improve as bandwidth increases. Further increase in network bandwidth may not necessarily pay off unless faster processors and better memory hierarchy are used. This is especially true in the case of the Option Red. Based on Fig. 15a, all ASCI machines get saturated in speed when the latency is reduced to 0.1 ms. From Fig. 15b, all machines cannot further improve in speed when the bandwidth increases beyond 100 GB/s.

#### *7.3.1 Combined Effects*

Below we scale both processor speed and network bandwidth simultaneously for both small and large workloads. The 3D diagrams in Fig. 16 show the combined effects. The speed range of the processors is limited from 1 to 16 Gflop/s. The bandwidth increases from 1.6 to 102.4 GB/s. These ranges are very close to the predicted processor speed and network bandwidth for the next decade.

For the small workload and low bandwidth, all ASCI machines increase in speed slightly with the use of faster processors. As the bandwidth increases, the performance surfaces become much steeper. The largest gap occurs when the bandwidth increases to 12.8 GB/s for a small workload. The message here is that simply increasing bandwidth alone without increasing processor speed will not pay off. The processor speed and network bandwidth should increase proportionally, as shown in Fig. 16.

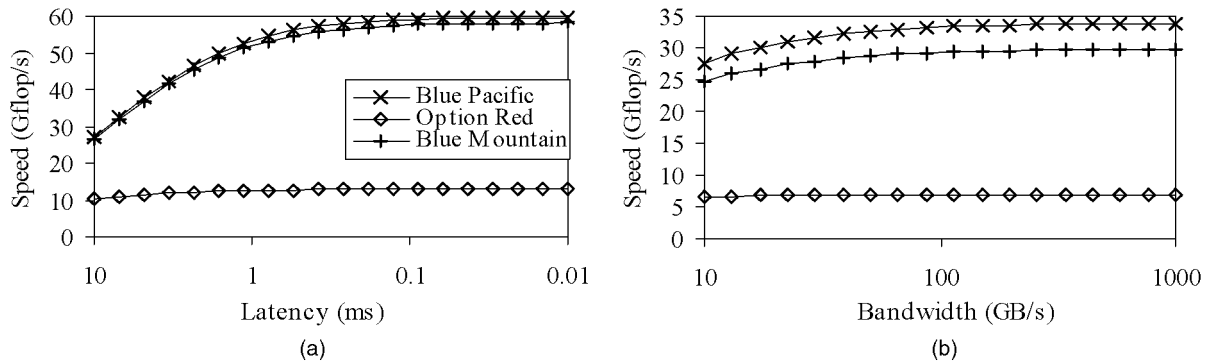



Fig. 15. Effects of communication latency and networkbandwidth on the ASCI machine performance.

For a large workload, all the speed surfaces on the right side increase steadily, even with a low bandwidth. With 1 Gflop/s processors and 1.6 GB/s link bandwidth, the Blue Mountain may achieve a 0.8 Tflop/s performance, about 25 percent of the peak speed in Table 2. The Blue Pacific may achieve 20 Tflop/s if the processor speed were scaled to 16 Gflop/s with an aggregate bandwidth of 102.4 GB/s. As Fig. 16b shows, the system speed is less sensitive to the increase of bandwidth alone. The Option Red is least sensitive to bandwidth increase for a large workload.

A machine running a small workload is more sensitive to bandwidth upgrade than scaling in processor speed. For a large workload, upgrading both processor speed and network bandwidth will enhance the performance linearly in both dimensions. However, the system speed becomes less sensitive to bandwidth increase when the processor speed becomes extremely high. In general, scaling along multiple resources dimensions simultaneously should be conducted in a balanced manner. Our scaling study has ignored the cost-effectiveness factor because it is out of the scope of this paper.

## 8 Related Work and Conclusions

We summarize below the major research findings of this scalability study. Then we discuss the limitations and possible generalization of our work. Finally, we make a number of suggestions for further research work to follow. Related works are also commented upon.

### 8.1 Summary of Research Findings

Among the three commercial MPPs, the SP2 shows the highest speed and efficiency. This is mainly attributed to the use of a well-designed switching network that is integrated with middleware support for clustering operations. By far, the SP2 has the best middleware support for *single system image* (SSI) in cluster-structured MPPs [16]. The T3D demonstrates the highest aggregate bandwidth with an impressive design in its memory hierarchy. The T3E is improved further in this aspect. Both caching and remote memory accesses have been accelerated in the T3D/T3E series.

The Paragon trails far behind in all performance attributes in our study. We found that the bottleneck comes mainly from the use of slow processors (i860 in Paragon and Pentium Pro in ASCI Red) and a long latency incurred with the execution of the NX or MPICH for passing messages. The mesh networks in the Intel MPP Series have been proven quite scalable in size, but not necessarily in performance. The long latency in Paragon is largely attributed to the use of low-level primitives or heavy weight processes for message passing.

This latency problem was solved in the SP2 and T3D by fine tuning of the MPI operations to better match with the target machine architecture. For communication-intensive programs (like the El-Stag), its speed performance is more sensitive to the bandwidth variation, while computation-intensive programs (like the HO-PD) are more sensitive to the magnitude of the message-passing latency. These two extreme program types are distinguished by the CCR measure shown in Table 4. The CCR value of a given program can be adjusted by algorithm redesign or by an intelligent compiler to yield a lower CCR.

Among the ASCI design options, the Blue Pacific has the potential to deliver the highest performance. Based on data released by IBM, it became the fastest supercomputer ever built up to late 1998. The Blue Pacific is quite scalable for having a cluster architecture which is supported by many SSI services such as single-entry point, global job management, single networking, etc. The strength of the Blue Pacific machine lies in its low latency, better software support, and many SSI services provided.

The Blue Mountain is projected to achieve the second best performance. Its DSM cluster architecture uses a high-bandwidth interconnect improved from the SGI/Cray Origin 2000. The DSM architecture limits its scalability, because remote memory update through page migration takes longer time than today's message-passing operations. However. the use of 1 Gflop/s SN1 processors helps the Blue Mountain accelerate floating-point operations.

The ASCI Red was designed in 1995 and implemented in 1996, two years ahead of the ASCI Blue machines, although the ASCI Red was ranked the world's fastest computer with a maximal speed of 1.338 Terflops. Our analysis shows that it may achieve the lowest STAP speed on today's ASCI Red configuration compared with the projected speed of the ASCI Blue machines. This was attributed to both hardware and software problems associated with the ASCI Red implementation in 1997.

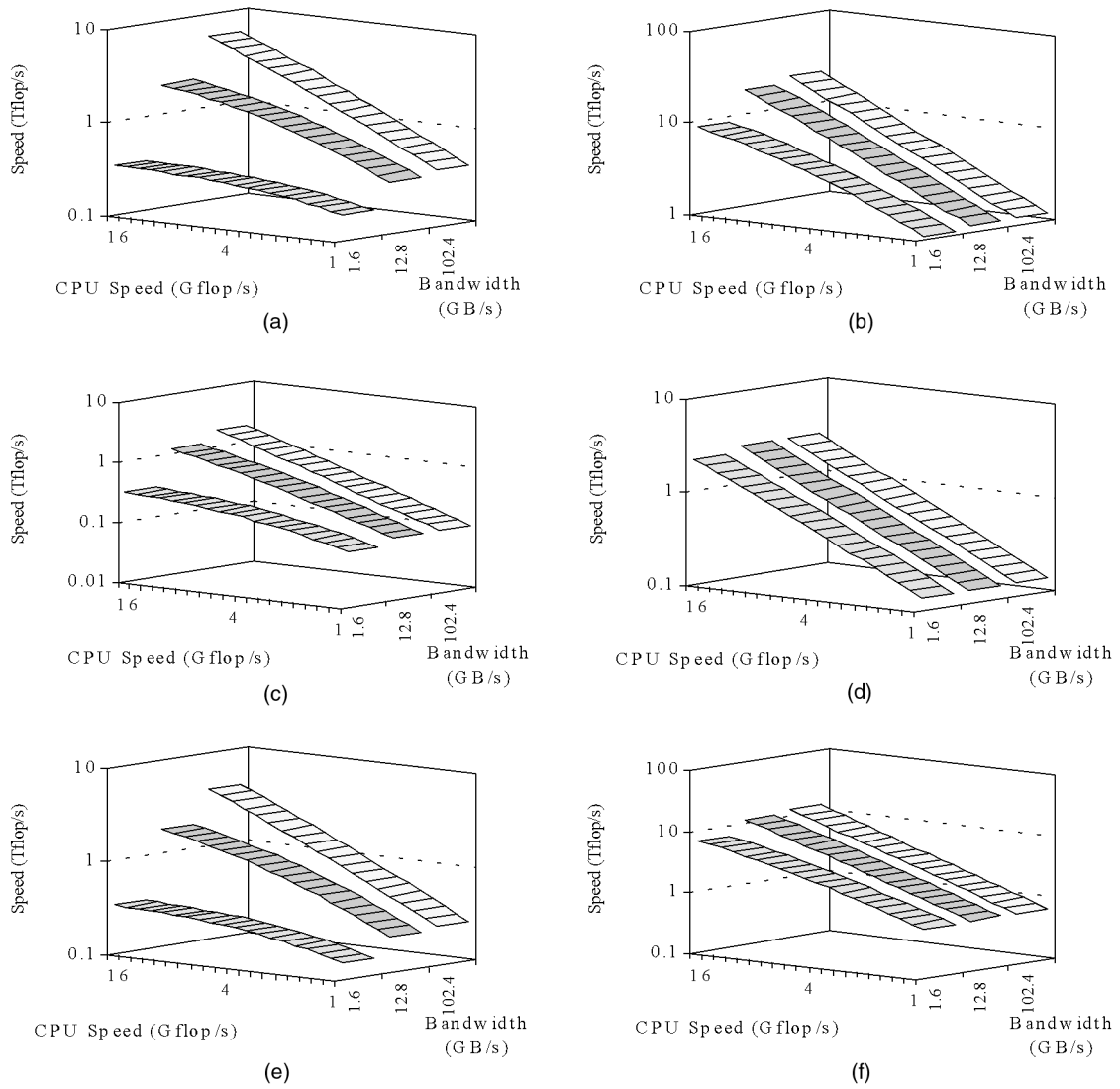



Fig. 16. Combined effects of scaling processor speed and communication bandwidth on the performance of ASCI machines.

It is not a surprise to project the trailing in ASCI Red speed due to the use of slower processors and older interconnect technology. Another drawback is on the software side. All Intel MPPs were implemented with third-party operating systems and message passing libraries, while both IBM and SGI/Cray have developed their own system software. The poor match between hardware and software has pulled down the performance of both Paragon and Option Red in succession. Of course, our speed projections are biased by STAP behavior. The ranking of the ASCI machines may change if different benchmarks are used to evaluate them in the future.

### 8.2 Limitations and Generalization

We discuss below MPP applications other than the STAP benchmark for signal processing. The above benchmark findings suggest that a balanced scalable design in both architectural resources and software support is the key to future success in massively parallel processing. Our results reinforce this rule of the thumb.

Our phase-parallel scaling model applies only to regularly structured coarse-grained SPMD programs. Our resource scaling method does not work with fine-grained SIMD (*single instruction stream and multiple data streams)* or MIMD (*multiple instruction streams and multiple data streams*) machines either [18]. This does pose some limitations in applying our scaling model.

The restriction is not as bad as it looks because most scientific codes running on existing MPPs choose the coarse-grained SPMD mode [18] instead of the fine-grained SIMD, MIMD, or coarse-grained MPMD (*multiple programs and multiple data streams* ) operations. Many real-life benchmarks have regular structures similar to the STAP benchmark. Therefore, our STAP benchmark experiments can be generalized to model the resource scaling effects of some regularly structured programs in the NAS [26], Linpack [12], C3I [24], SPLASH-2 [34], and Parkbench [14] benchmarks, all of which have coarse-grained characteristics.

All of the above benchmarks should be tested on current and future MPPs. The high-performance supercomputing

community has speculated the use of LAN-based multicomputer clusters [16], DSM multiprocessors, or even Internet-connected metacomputers to build future MPPs [10], [15], [16], [18]. However, neither DSM nor message passing is practical in handling massive parallelism because both page migration and MPI operations are still very slow today. Fast message passing and fast remote memory update must match the growth rate of processor speed before network-based machines can handle massive parallelism effectively.

Irregular problems pose another dimension of difficulty to evaluating the MPPs. Irregularly structured N-body problem has been studied by Singh et al. [29]. Their scaling experiments were conducted on the DASH multiprocessor. Both CCR and working-set size grow slowly when running a large problem on a larger machine. The lack of a single address space may substantially degrade the performance. Irregularity in program and data structures and unpredictable communication patterns demand a new scaling model that could be very different from our phase-parallel model.

The new scaling model must be designed to handle randomness in MPMD programs and to cope with the complex communication patterns encountered in distributed programs. Such a scaling scheme is even more difficult to formulate in a network-based cluster environment. The LogP model [10] was an initial attempt to handle randomly structured parallelism. Further research is needed to merge the strengths of the LogP model, the BSP model [32], and our phase-parallel model to deal with randomly structured parallelism. With today's long latency in passing messages, the new model may have to focus on coarse-grain MPMD programs.

### 8.3 Suggestions for Extended Work

Our STAP benchmark is MPI-coded. Our experiences suggest that users should always apply the most powerful, high-level MPI functions available instead of using a sequence of low-level primitives. For communication-intensive applications, such as the EL-Stag program in STAP, a high network/memory bandwidth is crucial to yield high performance. We suggest testing the STAP benchmark on clusters of workstations. However, the effects of long communication overhead in clusters are severer than those in today's MPPs. Therefore, the problems become more difficult to handle.

How to reduce the node granularity to fine grain posts an important challenge to increase the degree of parallelism in MPP or cluster applications. Hiding latencies within computations is desirable in many scientific or business computations. A possibility is to extend the phase-parallel model as such to overlap the computation phases with the interaction or communication phases. Clustering offers the potential to handle massive data parallelism in the future if the message passing overhead could be hidden within the computations. Besides, clusters offer higher availability in case of failure of a few nodes among thousands of nodes.

Our scaling model helps MPP designers determine the latency, bandwidth, processors, memory, networks, and I/O rate desired for specific applications. It can also be applied to optimize the partition of large workload among the MPP nodes. MPP users can apply our scaling model to reveal architectural bottlenecks and to trade off between computations and communications in the optimization of parallel applications. The processes of resource scaling, problem scaling, and cost scaling must be balanced to expect a truly scalable performance in future MPP systems or in large-scale clusters of computers.

## ACKNOWLEDGMENTS

This work was carried out by the co-authors at the University of Hong Kong during 1996-1998. We appreciate the valuable suggestions from the anonymous referees. We are indebted to Dr. Hai Jin for his help in redrawing some of the illustrations. The research was supported by Hong Kong Research Grants Council grants HKU 2/96C, HKU 7022/97E, HKU 548/96E, and HKU 7030/98E and by the development fund of the Area-of-Excellence in Information Technology from the University of Hong Kong in 1998.

## REFERENCES

[1] T. Agerwala, J.L. Martin, J.H. Mirza, D.C. Sadler, D.M. Dias, and M. Snir, "SP2 System Architecture," *IBM Systems J.*, vol. 34, no. 2, pp. 152-184, 1995.

[2] ASCI Blue Mountain, http://www.lanl.gov/Internal/projects/asci/bluemtn, Dec. 1997.

[3] ASCI Blue Pacific home page, http://www.llnl.gov/asci/platforms/bluepac, Dec. 1997.

[4] ASCI Option Red home page, http://mephisto.ca.sandia.gov/TFLOP/sc96, Dec. 1997.

[5] P. Bhat, Y. Lim, and V. Prasanna, "Scalable Portable Parallel Algorithms for STAP," *Proc. Adaptive Sensor Array Processing Workshop*, MIT Lincoln Lab., Mar. 1996.

[6] R. Bond, "Measuring Performance and Scalability Using Extended Versions of the STAP Processor Benchmarks," technical report, MIT Lincoln Lab., Dec. 1994.

[7] R.P. Brent, "The Parallel Evaluation of General Arithmetic Expressions," *J. ACM*, vol. 21, no. 2, pp. 201-206, 1972.

[8] D. Clark, "ASCI Pathforward: To 30 Tflops and Beyond," *IEEE Concurrency*, vol. 6, no.2, pp.13-15, Apr.-June, 1998.

[9] D. Crawford, "ASCI Academia Strategic Alliances Program: Research Interests in Computer Systems and Computational Science Infrastructure," Sandia National Lab., Dec. 1996.

[10] D.E. Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a Realistic Model of Parallel Computation," *Proc. ACM Symp. Principles and Practice of Parallel Programming*, pp. 1-12, 1993.

[11] J. Day, "Tutorial on Digital Adaptive Beamforming," *Proc. IEEE National Radar Conf.*, Martin Marietta Corp., Utica, N.Y., 22 Apr. 1993.

[12] J.J. Dongarra, "The Linpack Benchmark-Parallel Report," http://performance.netlib.org/performance/html/linpack-parallel.data.co10.html, 1996.

[13] J.L. Gustafson, "Reevaluating Amdahl's Law," *Comm. ACM*, vol. 31, pp. 532-533, 1988.

[14] R.W. Hockney and M. Berry, "Public International Benchmarks for Parallel Computers: PARKBENCH Committee Report No. 1," *Scientific Programming*, vol. 3, no. 2, pp. 101-146, 1994.

[15] C. Holt, M. Heinrich, J. Singh, E. Rothberg, and J. Hennessy, "The Effects of Latency, Occupancy, and Bandwidth in Distributed Shared Memory Multiprocessors," Technical Report CSL-TR-95-660, Computer Systems Lab., Stanford Univ., Jan. 1995.

[16] K. Hwang, E. Chow, C.L. Wang, H. Jin, and Z. Xu, "Designing SSI Clusters with Hierarchical Checkpointing and Single I/O Space," *IEEE Concurrency*, pp. 60-69, Jan.-Mar. 1999.

[17] K. Hwang, C.J. Wang, and C.-L. Wang, "Evaluating MPI Collective Communication on the SP2, T3D, and Paragon Multicomputers," *Proc. Third Int'l Symp. High-Performance Computer Architecture, (HPCA-3)*, pp. 106-116, San Antonio, Tex., 1-5 Feb. 1997.

[18] K. Hwang and Z. Xu, *Scalable Parallel Computing: Technology, Architecture, Programming*. New York: McGraw-Hill, 1998.

[19] K. Hwang, Z. Xu, and M. Arakawa, "Benchmark Evaluation of the IBM SP2 for Parallel Signal Processing," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 5, pp. 522-536, May 1996.
[20] J. Laudon and D.E. Lenoski, "The SGI Origin: A ccNUMA Highly Scalable Server," *Proc. 24th Int'l Symp. Computer Architecture,* pp. 241-251, June 1997.
[21] D.E. Lenoski, J. Laudon, T. Joe, and D. Nakahira, "The DASH Prototype: Logic Overhead and Performance," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 1, pp. 41-61, Jan. 1993.
[22] R.P. Martin, A.M. Vahdat, D.E. Culler, and T.E. Anderson, "Effects of Communication Latency, Overhead, and Bandwidth in a Cluster Architecture," *Proc. 24th Int'l Symp. Computer Architecture,* pp. 85-97, Denver, Colo., 2-4 June 1997.
[23] T.G. Mattson, D. Scott, and S. Wheat, "A TeraFLOPS Supercomputer in 1996: The ASCI TFLOPS System," *Proc. Sixth Int'l Parallel Processing Symp.,* pp. 84-93, 1996. Also available as http://www.cs.sandia.gov/ISUG97/papers/Mattson/OVERVIEW.html.
[24] R. Metzger, B. Voorst, L. Piere, R. Jha, W. Au, M. Amin, D. Cstanon, and V. Kumar, "The C3I Parallel Benchmark Suite - Introduction and Preliminary Results," *Proc. Supercomputing,* 1996.
[25] G.E. Moore, "Can Moore's Law Continue Indefinitely?" *Computerworld,* July 1996.
[26] W. Saphir, A. Woo, and M. Yarrow, "The NAS Parallel Benchmarks 2.1 Results," NASA Ames Research Center Report NAS-96-010, Aug. 1996.
[27] Silicon Graphics Inc., "Origin 200 and Origin 2000 Technical Report," Silicon Graphics Computer Systems, Mountain View, Calif., 1997.
[28] SGI/CRI, "Cray T3E Information," http://www.cray.com/products/systems/crayt3e/, 1997.
[29] J.P. Singh, J.L. Hennessy, and A. Gupta, "Implications of Hierarchical N-Body Methods for Multiprocessor Architectures," *ACM Trans. Computer Systems,* vol. 13, May 1995.
[30] D.G. Stunkel, B. Shea, M.G. Abali, C.A. Atkins, C.A. Bender, D.G. Grice, P. Hochschild, D.J. Joseph, B.J. Nathanson, R.A. Swetz, R.F. Stucke, M. Tsao, and P.R. Varker, "The SP2 High-Performance Switch," *IBM Systems J.,* vol. 34, no. 2, pp. 185-204, 1995.
[31] X.H. Sun and J. Zhu, "Performance Prediction, A Case Study Using a Scalable Shared-Virtual-Memory Machine," *IEEE Parallel and Distributed Technology,* pp. 36-49, Winter 1996.
[32] L.G. Valiant, "A Bridging Model for Parallel Computation," *Comm. ACM,* vol. 33, no. 8, pp. 103-111, Aug. 1990.
[33] C.J. Wang, C.-L. Wang, and K. Hwang, "STAP Benchmark Evaluation of the T3D, SP2, and Paragon," *Proc. 10th Int'l Conf. Parallel and Distributed Computing Systems, PDCS-97,* pp. 181-188, New Orleans, 1-3 Oct. 1997.
[34] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodology," *Proc. Inl'l Symp. Computer Architecture,* pp. 24-36, 1995.
[35] Z. Xu and K. Hwang, "Early Prediction of MPP Performance: SP2, T3D, and Paragon Experiences," *J. Parallel Computing,* vol. 22, pp. 917-942, Oct. 1996.
[36] Z. Xu and K. Hwang, "Modeling Communication Overhead: MPI and MPL Performance on the IBM SP2," *IEEE Parallel and Distributed Technology,* pp. 9-23, Mar. 1996.

**Kai Hwang** received the PhD degree in electrical engineering and computer science from the University of California at Berkeley in 1972. He is a professor of computer engineering at the University of Southern California. Prior to joining USC, he taught at Purdue University for many years. An IEEE fellow, Dr. Hwang specializes in computer architecture, digital arithmetic, parallel processing, and distributed computing. He has published more than 150 scientific papers and six books in computer science and engineering. His latest book, *Scalable Parallel Computing* (McGraw-Hill 1998), coauthored with Zhiwei Xu, covers the architecture and programming of scalable multiprocessors or multicomputer clusters. He served as a distinguished visitor of the IEEE Computer Society, on the ACM SigArch Board of Directors, and as the founding editor-in-chief of the *Journal of Parallel and Distributed Computing*. He chaired international conferences, ARITH-7 in 1985, ICPP 86, IPPS 96, and HPCA-4 in 1998. His current interests focus on fault tolerance and single system image in multicomputer clusters and integrated information technology for multiagent, Java, Internet, and multimedia applications.

**Choming Wang** was a PhD candidate in computer engineering at the University of Southern California, Los Angeles, where he received the MS degree in computer engineering in 1986. He received the BS degree in chemical engineering from the National Taiwan University in 1979. From 1986 to 1995, he worked as a software engineer at Siemens Pacesetter, Inc., Sylmar, California. His technical interest covers the areas of software engineering, parallel computing, and supercomputer applications.

**Cho-Li Wang** received his BS degree in computer science and information engineering from National Taiwan University in 1985. He earned his MS and PhD degrees in computer engineering from the University of Southern California in 1990 and 1995, respectively. He is currently an assistant professor of computer science at the University of Hong Kong. His research interests include high-speed networking, computer architectures, and software environment for distributed multimedia applications. He is a member of the IEEE.

**Zhiwei Xu** received his MS degree in electrical engineering from Purdue University in 1984 and his PhD degree in computer engineering from the University of Southern California in 1987. From 1987 to 1994, he taught at Rutgers University and New York Polytechnic University. He participated in the STAP/MPP benchmark projects led by Dr. Hwang at USC and HKU from 1994 to 1998. Presently, he is a professor and chief architect at the National Center for Intelligent Computing Systems (NCIC), Chinese Academy of Sciences, Beijing, China. He leads a design group at NCIC building a series of cluster-based superservers. His current interests lie in network-based computing, parallel programming, and supercomputing technology.